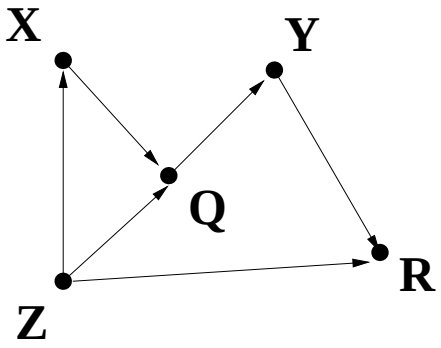
Digraphs

Digraphs, or directed graphs, are graphs where each edge has a direction.

Edges with direction are called arcs.



incident to- $x \to y$ **means x is incident to y**

incident from- $x \to y$ **means y is incident from x**

indegree- the number of arrow heads pointing toward a vertex

outdegree- the number of arrowheads coming out of a vertex

path- a sequence of arcs starting at x and ending at y in which no arc is repeated and each vertex in the sequence is incident to the next vertex

What vertices are incident to vertex X? Which are incident from vertex X? What is the indegree of vertex Q? What is the outdegree of vertex Z?

Examples:

Transportation:

Locations could be vertices, one-way streets could be arcs

Internet:

sources of information could be vertices, direction of the information transfer could be represented by arcs

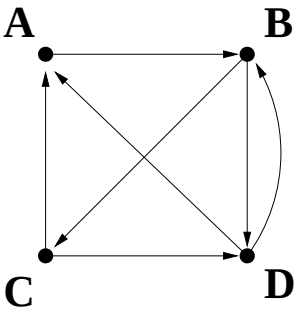
Pipelines:

cities could be vertices, direction of the flow in the pipes could be arcs

Chain of Command:

individuals could be vertices, arcs indicate chain of command, or who can give orders

Draw a digraph to represent the following situation: A likes B, but does not like C and D. B likes C and D but not A. C likes D and A. D likes B and A.



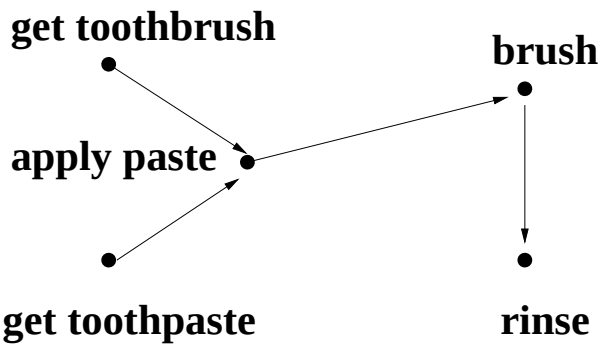
Digraphs can also represent tasks

Arrows indicate the order of task completion

Consider brushing your teeth:

- **get out the toothbrush**
- **get out the toothpaste**
- **apply toothpaste to toothbrush**
- **brush your teeth**
- **rinse out your mouth**

One possible digraph:



Using digraphs like this helps with SCHEDULING

Scheduling helps find efficient ways to organize a series of tasks

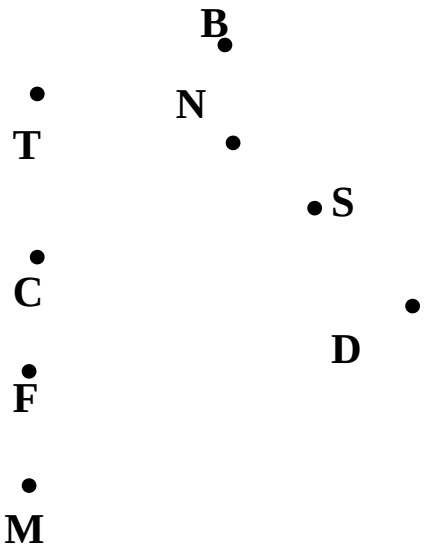
Definitions

Processors	**the "workers" that carry out the work** **denoted P_1, P_2 etc.**
Tasks	**the individual pieces of work** **denoted A, B, C etc.** **Note: a task is always carried out by a single processor**
Processing Times	**time required for one processor to finish a task** **denoted A(5)- it takes 5 hours, minutes, etc to do task A**
Precedence Relations	**describe tasks that must be completed before others can be started**
Independent Task	**has no restriction on when the task may be started**
Finishing time	**total length of project**
Critical time	**minimum time needed to complete project**

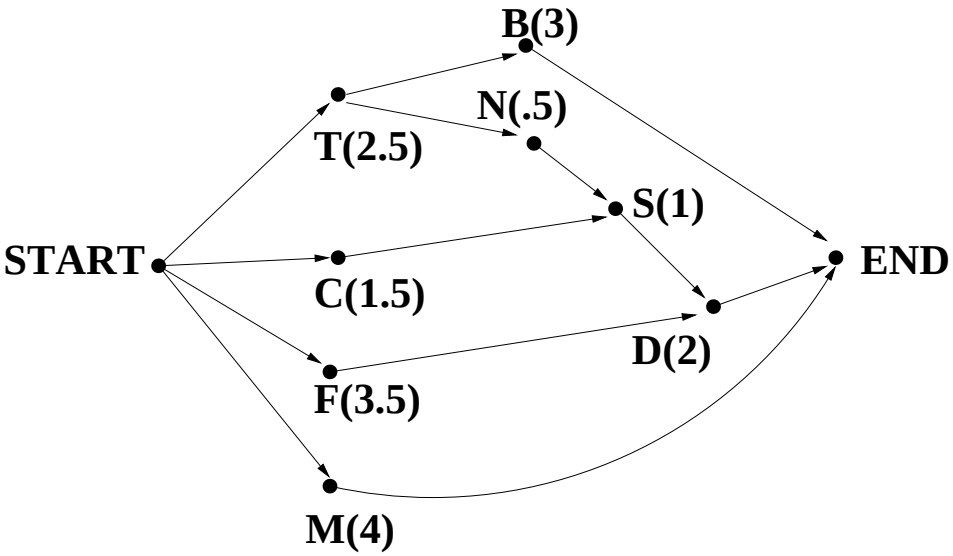
Example:

A ballroom is to be setup for a large wedding reception. The table lists the tasks, their notation, completion times and precedence relations.

Task	Symbol and time	Precedence relation
table setup	T(2.5)	T→N
tablecloth/napkins	N(0.5)	N → S
flower arrangement	F(3.5)	F → D
Crystal and China	C(1.5)	C → S
Set Table	S(1)	S → D
Table Decorations	D(2)	
Sound System	M(4)	
Bar	B(3)	T → B



Adding START and END vertices complete the picture:



If time, do a digraph for the steps involved in making a pasta dinner.

Make a spaghetti dinner for friends:

A digraph gives tasks and shows precedence relations

How to schedule the tasks?

Start by making a list of tasks, preferably in order of priority.

List is called Priority List

A random priority list:

T(2.5), B(3), M(4), N(0.5), F(3.5), S(1), D(2), C(1.5)

There are M! number of priority lists, where M is the number of tasks.

Once we have a priority list, we can begin scheduling.

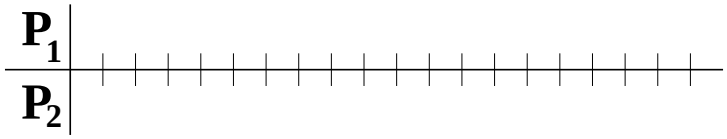
Pay attention to processors and task order

Schedule the tasks in order of priority, and task order, and processor availability

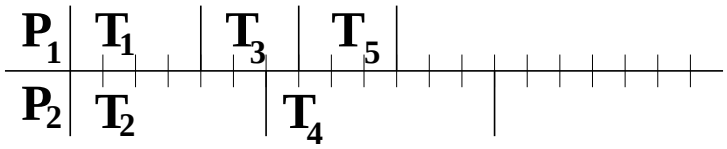
Representing the processors:

List the processors on a time line, to make it easy to see how long tasks take.

Example:



Write the tasks according to which processor is doing them and keep track of processing times:

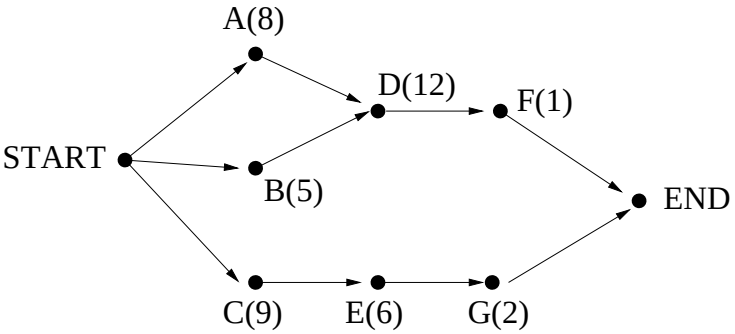


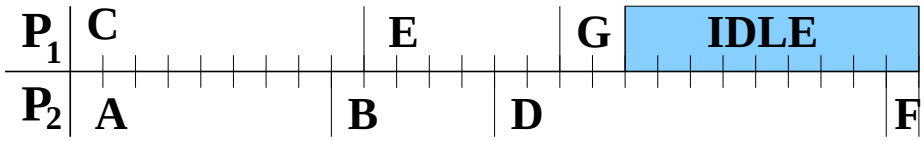
If each mark represents one hour, how long does it take to complete task 1 (T_1)?

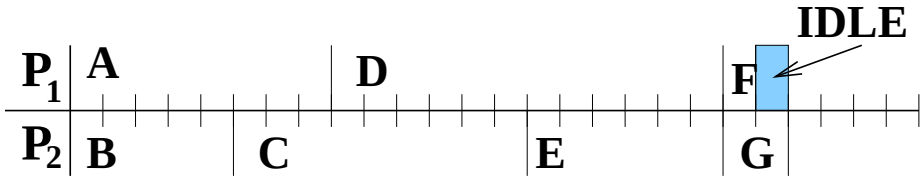
Which processor finishes first?

If T_4 could not be started until T_3 was completed, would this schedule work?

Consider the following digraph-







Which schedule is better? Why?